Offline Reinforcement Learning With Reverse Diffusion Guide Policy

Jiazhi Zhang, *Graduate Student Member, IEEE*, Yuhu Cheng, *Member, IEEE*, Shuo Cao, *Graduate Student Member, IEEE*, and Xuesong Wang, *Member, IEEE*

***Abstract*—Offline reinforcement learning (ORL) learns policy from a static dataset without further interaction with the environment, which holds significant promise in industrial control systems characterized by inefficient online interaction and inherent safety concerns. To mitigate the extrapolation error induced by distribution shift, it is essential for ORL to constrain the learned policy to perform actions within the support set of behavior policy. Existing methods fail to represent the behavior policy properly and typically tend to prefer actions with higher densities within the support set, resulting suboptimal learned policy. This article proposes a novel ORL method which represents the behavior policy with a diffusion model and trains a reverse diffusion guide policy to instruct the pretrained diffusion model in generating actions. The diffusion model exhibits stable training and strong distribution expression ability, and the reverse diffusion guide policy can effectively explore the entire support set to help generate the optimal action. When facing low-quality datasets, a trainable perturbation can be further added to the generated action to help the learned policy escape the performance limitation of behavior policy. Experimental results on D4RL Gym-MuJoCo benchmark demonstrate the effectiveness of the proposed method, surpassing several state-of-the-art ORL methods.**

***Index Terms*—Diffusion model, offline reinforcement learning (ORL), perturbation model, policy constraint, reverse diffusion guide policy (RDGP).**

I. Introduction

Reinforcement learning (RL) can effectively solve sequential decision problems through trial and error [1], which has achieved great success in games [2]. Nevertheless, the manner of online interaction with environment limits the further application of RL in the real-world industrial control systems [3]. In some real industrial scenarios, the online interaction is inefficient and may have security risks [4]. For instance, training a robot to walk in the real world may take several weeks, necessitating human intervention for a state reset whenever it falls [5]. In the context of autonomous driving, the intelligent vehicle may pose a risk to individuals or other vehicles around it during the learning process [6], [7]. Recent practical application of RL in the real world involves obtaining a well-initialized policy from existing datasets followed by fast online fine-tuning with limited interactions [8]. The initial phase, which learns a good policy from a static dataset without interaction with the environment, is referred to as offline reinforcement learning (ORL) [3]. The emergence of ORL provides more possibilities for the application of RL in real-world industrial control systems [9].

However, it is not trivial to learn a good policy only from a static dataset, as the dataset is hard to capture all the scenarios in the real applications. Directly applying online off-policy RL methods to offline setting typically fails to yield satisfactory results [10]. The performance of RL method relies on the precise estimation of value function [11]. During the process of value function estimation, actions sampled from the learned policy are utilized for bootstrap backup. However, when the sampled action does not appear in the static dataset, its value will be erroneously overestimated [12]. While this issue is also present in online off-policy RL method, in online settings, the true value of the action can be observed through interaction with the environment, allowing for error correction. However, in offline settings, this overestimation will be further amplified during the bootstrapping process. The above problems are summarized as extrapolation error caused by distribution shift in ORL. To mitigate the extrapolation error, it is essential to constrain the learned policy to perform actions within the support set of behavior policy [13]. Existing approaches can be broadly divided into three categories.

1) *Policy constraint* which constrains the learned policy to be close to the behavior policy, diminishing the likelihood of sampling out-of-distribution (OOD) actions during the bootstrapping process [14].
2) *Value function penalty* which punishes the learned value function by assigning low values to OOD actions [15]. Some studies have demonstrated that such method actually implicitly enforces policy constraint [16].
3) *In-sample* which avoids sampling actions according to the learned policy and instead exclusively utilizes samples from the dataset for value function estimation [17].

Manuscript received 27 December 2023; revised 28 April 2024; accepted 21 May 2024. Date of publication 27 June 2024; date of current version 7 October 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62373364 and Grant 62176259, and in part by the Key Research and Development Program of Jiangsu Province under Grant BE2022095. Paper no. TII-23-5242.
(Corresponding author: Xuesong Wang.)

The authors are with the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China (e-mail: zjzcumt@163.com; chengyuhu@163.com; scao@cumt.edu.cn; wangxuesongcumt@163.com).
Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2024.3413308.

Digital Object Identifier 10.1109/TII.2024.3413308

The policy is ultimately extracted through the weighted behavior cloning [18].

This article concentrates on the first category, policy constraint methods. Classical approaches involve measuring the distance between policy distributions through divergence and subsequently constraining this distance as a regularization term during policy training [19]. However, constraining the distance between policies essentially encourages the overall closeness between the two distributions, and it does not precisely restrict the learned policy to selecting actions only within the support set of behavior policy. Furthermore, this constraint method also compels the density distribution of the learned policy to closely match that of the behavior policy, which can be overly strict in ORL [20]. Consider a state where the behavior policy has a higher probability of selecting an action with lower reward. Regularizing through distance metrics would make the learned policy also choose this suboptimal action with a higher probability. However, in reality, the learned policy should select the action within the support set of behavior policy that yields higher reward, even if the probability density of this action in the behavior policy is lower.

Another approach to policy constraint involves directly modeling the behavior policy and then sampling multiple actions from it, selecting the action that maximizes the reward for execution [10], [21]. However, the sampling operation is also influenced by the density of behavior policy, and only when an infinite number of actions are sampled each time can the optimal action within the support set of behavior policy be consistently obtained. Nevertheless, conducting an infinite number of samples is impractical, then a key question arises, *How can we obtain the optimal action within the support set of behavior policy and avoid multiple sampling?* To answer this question, it is first necessary to understand how to properly represent the behavior policy, as only when the behavior policy is precisely estimated can it be ensured that the sampled actions are within the support set of behavior policy. The conventional approaches model the policy as a Gaussian distribution, which is generally effective in online situations [5]. However, in offline scenarios, the static datasets may be collected by human experts or mixed policies, often exhibiting complex multimodal characteristics that a Gaussian distribution cannot effectively represent [22]. Generative adversarial network possess powerful distribution modeling capabilities, but its unstable adversarial learning mechanism has long been criticized [23]. Variational autoencoders struggle to select an appropriate variational posterior that maintains strong expressive power while being easy to optimize [24].

Drawing inspiration from the remarkable success of diffusion models in the field of image generation [25], [26], we propose employing a diffusion model to represent the behavior policy. We believe that diffusion models hold promise for effectively representing multimodal behavior policy, because the dimensionality of actions in control tasks is typically much smaller than that of images in visual tasks. A bandit experiment used to visually demonstrate the effectiveness of using diffusion models to represent multimodal behavior policy (action distribution) can be found in the Supplementary Material. The predefined positive diffusion process maps empirical data to a standard Gaussian by gradually adding noise, whereas the trained inverse diffusion process restores the standard Gaussian to the original data distribution by gradually denoising. The incremental noise addition and denoising process endows the diffusion model with powerful distributional representation capabilities [27], [28]. By maximizing the evidence lower bound, a concise and clear surrogate optimization objective can be obtained. In addition, without employing an adversarial learning mechanism, the diffusion model employs a much more stable training process compared with GANs [29].

The optimized inverse diffusion process provides a mapping from the standard Gaussian to the behavior policy distribution, and any point sampled from the standard Gaussian corresponds to actions within the support set of behavior policy. To find the optimal action within the support set, we aim to locate its corresponding sampling point in the standard Gaussian. Specifically, we construct a reverse diffusion guide policy (RDGP) to map from states to optimal sampling points. Through this guidance process, we can flexibly select actions from the support set without being influenced by the density distribution of behavior policy, obtaining the optimal action in a single sampling step. In addition, we consider a broader scenario where the quality of the datasets is poor (the globally optimal action may not exist within the support set of behavior policy). In this case, a trainable perturbation can be further added to the sampled actions to enhance the OOD generalization. The main contributions of this article can be summarized as follows.

1) Introducing a method to represent behavior policy with diffusion (RBPD) model. Diffusion model can effectively capture the multimodal distribution of the empirical dataset, ensuring the accurate generation of actions within the support set of behavior policy.
2) Building upon RBPD, proposing a novel ORL method based on RDGP. This method avoids the multiple sampling processes in previous approaches, enabling the direct acquisition of the optimal action within the support set of behavior policy through guided sampling. When facing low-quality datasets, a trainable perturbation can be further added to enhance the OOD generalization.
3) Conducting experimental comparisons of the proposed RDGP with various state-of-the-art ORL methods on the D4RL Gym-MuJoCo benchmark. The results demonstrate that RDGP can effectively adapt to tasks with different dataset qualities and achieve optimal performance on most tasks with minimal hyperparameter tuning or even without hyperparameter tuning.

The rest of this article is organized as follows. Section II gives the introduction of relevant background knowledge, including RL and ORL. Section III is divided into two parts, the first part introduces RBPD, and the second part presents RDGP. Section IV conducts experiments and provides an analysis of proposed methods, including comparisons with various classical ORL methods. Finally, Section V concludes this article and summarizes the findings and provides a discussion of potential future work.

II. Preliminaries

A. Reinforcement Learning

RL considers a Markov decision process $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mu_0, \gamma, r\}$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state space and the action space, respectively, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ denotes the state transition probability, μ_0 represents the initial state distribution, $\gamma \in (0, 1]$ is the discount factor, and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ denotes the reward function. A policy $\pi :\ \mathcal{S} \to P(\mathcal{A})$ denotes a mapping from a given state to the probability distribution over the action space. The goal of RL is to obtain the optimal policy π^* that maximizes the discounted cumulative reward as follows:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] \tag{1}$$

where $\tau = (s_0, a_0, s_1, a_1, \cdots)$ denotes a randomly trajectory generated according to policy π, $s_0 \sim \mu_0$, $a_t \sim \pi(\cdot|s_t)$, and $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$.

The action value function $Q^\pi(s, a)$ (also known as Q-value function) represents the expected discounted cumulative reward obtained by following policy π from state s and action a, it serves as the fixed point of Bellman operator $\mathcal{T}^\pi$ as follows:

$$\left(\mathcal{T}^\pi Q\right)(s, a) = r(s, a) + \gamma \mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a), a'\sim\pi(s')}\left[Q(s', a')\right]. \tag{2}$$

Off-policy actor-critic methods achieve π^* by alternating between policy evaluation and policy improvement, the policy evaluation estimates action-value function $Q^\pi(s, a)$ according to $\mathcal{T}^\pi$, the policy improvement updates policy π to maximize $Q^\pi(s, a)$.

B. Offline Reinforcement Learning

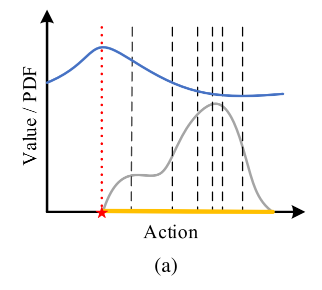
ORL methods assume no interaction with the environment and only have a static dataset $\mathcal{D} = \{(s, a, r, s')\}$. $\mathcal{D}$ is collected by some unknown behavior policy π_b, which may be a single policy or a mixed policy with multimodal distribution. For instance, in the context of an autonomous driving task, different datasets may be collected by different drivers, corresponding to different behavior policies. Consequently, a mixed dataset that incorporates driving experiences from different drivers would correspond to a mixed behavior policy. In this case, owing to variations in driver skill, different actions may be taken in the same state, thus resulting a multimodal behavior policy. Since interaction with the environment is forbidden, offline actor-critic methods employ the empirical Bellman operator $\bar{\mathcal{T}}^\pi$ instead of $\mathcal{T}^\pi$ for updating Q values as follows:

$$Q_{k+1} = \arg\min_{Q} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\left(\left(\bar{\mathcal{T}}^{\pi_k} Q_k\right)(s, a) - Q(s, a)\right)^2\right] \tag{3}$$

$$\pi_{k+1} = \arg\max_{\pi} \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi(s)}\left[Q_{k+1}(s, a)\right] \tag{4}$$

where

$$\left(\bar{\mathcal{T}}^{\pi_k} Q\right)(s, a) = r(s, a) + \gamma \mathbb{E}_{s'\sim\mathcal{D}, a'\sim\pi_k(s')}\left[Q(s', a')\right]. \tag{5}$$



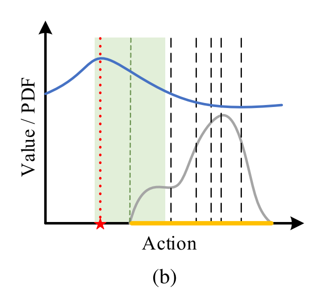


Fig. 1. Core idea of RDGP. (a) Optimal action lies within the support set of behavior policy. (b) Optimal action lies outside the support set of behavior policy.

Notice that a' is sampled from the learned policy π_k, it might not appear in $\mathcal{D}$ (especially when π_k is much different from π_b), resulting erroneously overestimated Q-value. ORL can perform a' in the environment and observing its true value to correct the estimation biases, but in offline setting, interaction processes are forbidden. A standard approach is to impose constraints during the policy updating process, restricting the learned policy to select actions only within the support set of behavior policy as follows:

$$\pi_{k+1} = \arg\max_{\pi} \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi(\cdot|s)}\left[Q_{k+1}(s, a)\right]$$
$$s.t.\quad \text{support}(\pi) \subset \text{support}(\pi_b). \tag{6}$$

To achieve this, the batch-constrained Q-learning (BCQ) employs a variational autoencoder to estimate the behavior policy and samples candidate actions exclusively from the estimated behavior policy. It introduces a novel Q-value updating rule known as the expectation-maximization operator as follows:

$$\left(\mathcal{T}^K_{\tilde{\pi}_b} Q\right)(s, a)$$
$$= r(s, a) + \gamma \mathbb{E}_{s'\sim\mathcal{D}, \{a'_i\}^K\sim\tilde{\pi}_b(\cdot|s')}\left[\max_{a'\in\{a'_i\}^K} Q(s', a')\right] \tag{7}$$

where $\tilde{\pi}_b$ denotes the behavior policy estimated by a conditional variational autoencoder, and K denotes the number of candidate actions sampled from $\tilde{\pi}_b$. Since the sampling process is influenced by the density distribution of behavior policy, the expectation-maximization operator converges to the optimal value only when $K \to \infty$.

III. ORL With Diffusion Guide Policy

First, we illustrate the core idea of our method through two schematic diagrams. As shown in Fig. 1(a), the blue line represents the optimal Q-value function for a certain state; the gray line represents the probability density function of the behavior policy in that state; the yellow region represents the support set of behavior policy; the black dashed line represents the process of sampling actions from the behavior policy, naturally favoring regions with higher probability density; and the red star represents the optimal action, corresponding to the maximum Q-value in current state. Traditional methods involve multiple samplings from the behavior policy, and the action with maximum Q-value among these samples is chosen for execution. When the action space is continuous or the optimal action lies

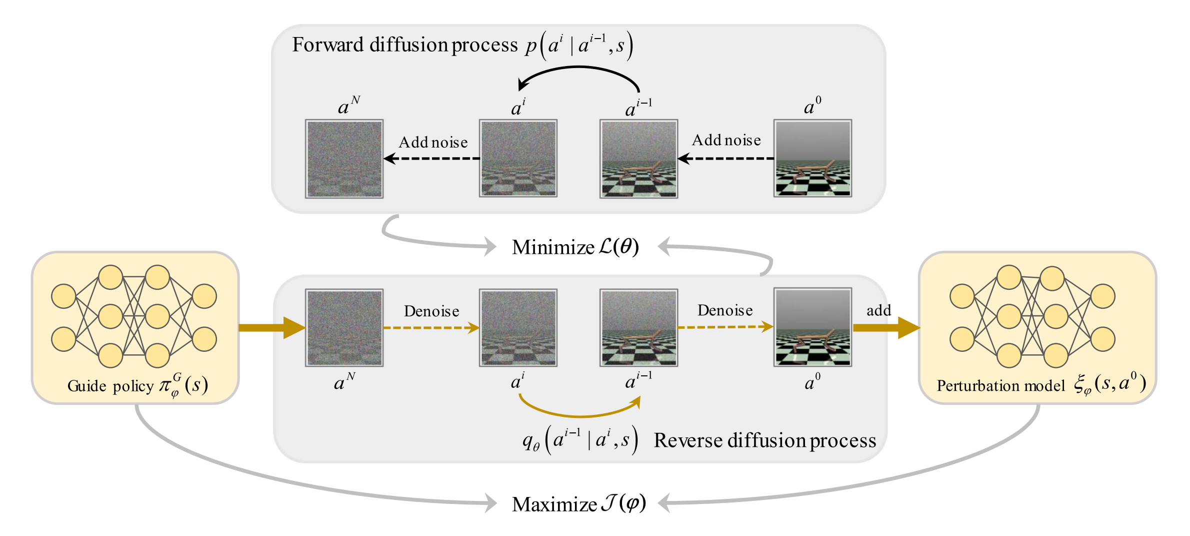


Fig. 2. ORL based on RDGP.

in a region of low probability density in the behavior policy, it becomes challenging to sample the optimal action using this approach ("discrete" sampling processes struggle to precisely sample the optimal action). Only with an infinite number of samples can candidate actions cover the entire support set of behavior policy to achieve optimal action selection. However, in practical operations, conducting an infinite number of samples is not feasible. In response, we shift our approach and directly utilize Q-values to guide the sampling process, flexibly exploring the entire support set and directly sampling the action with maximum Q-value. When the optimal action lies within the support set of behavior policy, the above method can effectively obtain the optimal action. However, as shown in Fig. 1(b), when the optimal action lies outside the support set of behavior policy, the action with maximum Q-value within the support set may still be a suboptimal action [green dashed line in Fig. 1(b)]. In this case, we further introduce perturbations to the actions sampled through guidance [green region in Fig. 1(b)], enabling them to generalize to areas outside the support set.

Specifically, we propose a novel ORL method based on RDGP as illustrated in Fig. 2, to implement the aforementioned idea. First, we use a diffusion model to represent the behavior policy. Leveraging the powerful distribution representation capabilities of diffusion model, we can accurately sample actions from the support set of behavior policy. Next, we train a RDGP to guide the sampling process, obtaining the action with maximum Q-value within the support set through a single sampling process. Finally, when the optimal action lies outside the support set (indicating poor quality of the static datasets), we introduce a trainable perturbation to the sampled action to enhance its generalization ability. The following sections provide detailed explanations of how to represent the behavior policy with a diffusion model, and how to train the RDGP and the perturbation model.

A. RBPD Model

Diffusion model is a powerful class of generative model, which has been widely applied in natural language processing and image generation [30]. Here, we employ a diffusion model to learn behavior policy π_b from dataset $\mathcal{D}$. As shown in Fig. 2, the diffusion model can be divided into the forward diffusion process and the reverse diffusion process. The forward diffusion process can be viewed as a predefined noise-adding process. It starts by sampling state-action pairs (s, a) from the offline dataset, then takes s as a condition and a as the initial action a^0. Gradually, Gaussian noise is added to a^0 until it becomes entirely Gaussian noise a^N as follows:

$$p\left(a^{1:N}|a^0, s\right) = \prod_{i=1}^{N} p\left(a^i|a^{i-1}, s\right) \tag{8}$$

$$p\left(a^i|a^{i-1}, s\right) = \mathcal{N}\left(a^i; \sqrt{1-\beta_i}a^{i-1}, \beta_i I\right) \tag{9}$$

where i denotes the time step in the diffusion process, and β_i represents the noise coefficient. Let $\alpha_i = 1 - \beta_i$ and $\bar{\alpha}_i = \prod_{n=1}^{i} \alpha_i$, then $p(a^i|a^0, s)$ can be recursively derived through repeated applications of the reparameterization trick [31]. For an arbitrary sample $a^i \sim p(a^i|a^0, s)$, it can be rewritten as follows:

$$\begin{aligned}
a^i &= \sqrt{1-\beta_i}a^{i-1} + \sqrt{\beta_i}\hat{\epsilon}_{i-1} \\
&= \sqrt{\alpha_i}a^{i-1} + \sqrt{1-\alpha_i}\hat{\epsilon}_{i-1} \\
&= \sqrt{\alpha_i}(\sqrt{\alpha_{i-1}}a^{i-2} + \sqrt{1-\alpha_{i-1}}\hat{\epsilon}_{i-2}) + \sqrt{1-\alpha_i}\hat{\epsilon}_{i-1} \\
&= \sqrt{\alpha_i\alpha_{i-1}}a^{i-2} + \sqrt{\alpha_i - \alpha_i\alpha_{i-1}}\hat{\epsilon}_{i-2} + \sqrt{1-\alpha_i}\hat{\epsilon}_{i-1} \\
&= \sqrt{\alpha_i\alpha_{i-1}}a^{i-2} + \sqrt{1-\alpha_i\alpha_{i-1}}\epsilon_{i-2} \\
&= \ldots \\
&= \sqrt{\prod_{n=1}^{i}\alpha_i}a^0 + \sqrt{1-\prod_{n=1}^{i}\alpha_i}\epsilon_0 \\
&= \sqrt{\bar{\alpha}_i}a^0 + \sqrt{1-\bar{\alpha}_i}\epsilon_0 \\
&\sim \mathcal{N}(a^i; \sqrt{\bar{\alpha}_i}a^0, (1-\bar{\alpha}_i)I)
\end{aligned} \tag{10}$$

where $\{\epsilon_i, \hat{\epsilon}_i\}_{i=0}^{N} \sim^{\text{iid}} \mathcal{N}(0, I)$, "iid" represents independent and identically distributed. Thus, $p(a^i|a^0, s) = \mathcal{N}(a^i; \sqrt{\bar{\alpha}_i}a^0, (1-\bar{\alpha}_i)I)$.

On the other hand, the reverse diffusion process can be viewed as a trainable denoising process $q_\theta(a^{i-1}|a^i, s)$. Its goal is to gradually denoise a^N and restore it to the corresponding initial action a^0 as follows:

$$q_\theta\left(a^{0:N}|s\right) = q\left(a^N\right)\prod_{i=1}^{N} q_\theta\left(a^{i-1}|a^i, s\right) \tag{11}$$

$$q_\theta\left(a^{i-1}|a^i, s\right) = \mathcal{N}\left(a^{i-1}; \mu_\theta(a^i, s, i), \Sigma_\theta(a^i, s, i)\right) \tag{12}$$

where $\mu_\theta(a^i, s, i) = \frac{1}{\sqrt{\alpha_i}}\left(a^i - \frac{1-\alpha_i}{\sqrt{1-\bar{\alpha}_i}}\epsilon_\theta(a^i, s, i)\right)$ and $\Sigma_\theta(a^i, s, i) = \beta_i I$. The reverse diffusion process is optimized by maximizing the evidence lower bound $\mathbb{E}_p\left[\ln\frac{q_\theta(a^{0:N}|s)}{p(a^{1:N}|a^0,s)}\right]$. According to denoising diffusion probabilistic models [29], optimizing the reverse diffusion process is equivalent to optimizing the noise model ϵ_θ, the surrogate loss function can be expressed as follows:

$$\mathcal{L}(\theta) = \mathbb{E}_{i\sim\mathcal{U},\epsilon\sim\mathcal{N}(0,I),(s,a)\sim\mathcal{D}}\left[\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_i}a + \sqrt{1-\bar{\alpha}_i}\epsilon, s, i\right)\right\|^2\right] \tag{13}$$

where $\mathcal{U}$ is defined as a uniform distribution on the discrete set $\{1, \ldots, N\}$. After training, each sampling of $a^N \sim \mathcal{N}(0, I)$, followed by iterative denoising based on the reverse diffusion process $q_\theta(a^{i-1}|a^i, s)$, results in a final a^0 that conforms to the distribution of the dataset. In other words, the reverse diffusion process represents the behavior policy as follows:

$$\tilde{\pi}_\theta^b(a|s) = q_\theta\left(a^{0:N}|s\right) = \mathcal{N}\left(a^N; 0, I\right)\prod_{i=1}^{N} q_\theta\left(a^{i-1}|a^i, s\right). \tag{14}$$

B. RDGP Training

The reverse diffusion process provides a mapping from the standard Gaussian distribution to the behavior policy distribution. Any a^N sampled from the standard Gaussian distribution corresponds to an action a^0 within the support set of behavior policy. To obtain the optimal action $\bar{a}^0$ within the support set, we consider training a RDGP to obtain its corresponding $\bar{a}^N$. The RDGP is denoted as a deterministic policy $\pi_\varphi^G(s)$. As shown in Fig. 2, when generating actions through the reverse diffusion process, we no longer sample a^N from the standard Gaussian distribution. Instead, we use the output of $\pi_\varphi^G(s)$ as a substitute for the sampled result. Then, the final executed action is obtained by iteratively denoising according to the reverse diffusion process $q_\theta(a^{i-1}|a^i, s)$. In this way, the learned policy can be represented as follows:

$$\pi_\varphi(s) = \pi_\varphi^G(s)\prod_{i=1}^{N} q_\theta\left(a^{i-1}|a^i, s\right) \tag{15}$$

where the learned policy $\pi_{\varphi,\theta}(s)$ is simplified as $\pi_\varphi(s)$, since the parameters θ of the reverse diffusion process are fixed after pretraining and are not optimized during the subsequent optimization process. The objective of the learned policy is to output the action within the support set of behavior policy that maximizes the expected return in the current state. As the reverse diffusion process ensures that the output action is always within the support set of behavior policy, the goal is to maximize the return. According to the deep deterministic policy gradient algorithm [32], the objective function for maximizing the return can be written as follows:

Algorithm 1: RDGP.

Initialize: Network parameters θ, υ_1, υ_2, and φ, and target network parameters $(\bar{\upsilon}_1, \bar{\upsilon}_2, \bar{\varphi}) \leftarrow (\upsilon_1, \upsilon_2, \varphi)$
Represent behavior policy with diffusion model (RBPD)
for each gradient step **do**
  Sample transition mini-batch $\mathcal{B} = \{(s, a, r, s')\} \sim \mathcal{D}$
  Update θ by minimizing $\mathcal{L}(\theta)$ in (13)
end for
Reverse Diffusion Guide Policy Training
for each gradient step **do**
  Sample transition mini-batch $\mathcal{B} = \{(s, a, r, s')\} \sim \mathcal{D}$
  Sample $\hat{a}^0 \sim \pi_{\bar{\varphi}}(s')$ by iteratively denoising according to (15)
  Update υ_1 and υ_2 by minimizing $\mathcal{L}(\upsilon_k)$ in (20)
  Sample $a^0 \sim \pi_\varphi(s)$ by iteratively denoising according to (15)
  Update φ by maximizing $\mathcal{J}(\varphi)$ in (19)
  Update $\bar{\upsilon}_1$, $\bar{\upsilon}_2$, and $\bar{\varphi}$ by Polyak averaging according to (18)
end for

$$\mathcal{J}^G(\varphi) = \mathbb{E}_{s\sim\mathcal{D},a^0\sim\pi_\varphi(s)}\left[Q_\upsilon\left(s, a^0\right)\right] \tag{16}$$

where Q_υ represents the value function network, and the clipping double Q-learning technique from TD3 [33] is applied to update here. Specifically, we parameterize two value function networks Q_{υ_1} and Q_{υ_2} [either Q_{υ_1} or Q_{υ_2} can be used in (16)], and optimize them by minimizing the following loss function:

$$\mathcal{L}^G(\upsilon_k) = \mathbb{E}_{(s,a,s')\sim\mathcal{D},\hat{a}^0\sim\pi_{\bar{\varphi}}(s')}\left[\left(Q_{\upsilon_k}(s, a) - \left(r + \gamma\min_{k=1,2} Q_{\bar{\upsilon}_k}\left(s', \hat{a}^0\right)\right)\right)^2\right],\ k = 1, 2 \tag{17}$$

where $\pi_{\bar{\varphi}}$, $Q_{\bar{\upsilon}_1}$, and $Q_{\bar{\upsilon}_2}$ represent the target networks corresponding to π_φ, Q_{υ_1}, and Q_{υ_2}, respectively. The use of target networks in the bootstrapping update process helps stabilize the learning process, and the target network parameters are updated according to Polyak averaging as follows:

$$\bar{\varphi} \leftarrow \eta\varphi + (1-\eta)\bar{\varphi},$$
$$\bar{\upsilon}_k \leftarrow \eta\upsilon_k + (1-\eta)\bar{\upsilon}_k \tag{18}$$

where η represents the target network update rate.

As shown in Fig. 1(b), when the optimal action lies outside the support set of behavior policy, an additional perturbation can be added to a^0 to search for the optimal action. Specifically, we construct a perturbation model $\xi_\varphi(s, a^0)$ to determine the specific size of the perturbation. The perturbed action is denoted as $a^0 + \xi_\varphi(s, a^0)$. Then, the optimization objective for the perturbation model and the learned policy can be obtained as follows:

$$\mathcal{J}(\varphi) = \mathbb{E}_{s\sim\mathcal{D},a^0\sim\pi_\varphi(s)}\left[Q_v\left(s, a^0 + \xi_\varphi\left(s, a^0\right)\right)\right] \quad (19)$$

Similarly, the loss function for updating the value function in this case can be obtained as follows:

$$\mathcal{L}(v_k) = \mathbb{E}_{(s,a,s')\sim\mathcal{D},\hat{a}^0\sim\pi_{\bar{\varphi}}(s')}\left[\left(Q_{v_k}(s,a) - \left(r + \gamma \min_{k=1,2} Q_{\bar{v}_k}\left(s', \hat{a}^0 + \xi_{\bar{\varphi}}\left(s', \hat{a}^0\right)\right)\right)\right)^2\right] \quad (20)$$

where $\xi_{\bar{\varphi}}$ represents the target network corresponding to ξ_φ and $k = 1, 2$. It is worth noting that while the perturbation model can enhance the OOD generalization of the learned policy, excessive perturbation can also introduce distribution shift. In practical implementation, RDGP always adds a perturbation and restricts the perturbation range to $[-\vartheta, \vartheta]$, where ϑ is the perturbation threshold. In the Supplementary Material, we investigated the impact of ϑ on the performance of the RDGP for datasets with different qualities and provided a general guideline for selecting ϑ. The implementation of RDGP is summarized in Algorithm 1.

IV. Experiment

In this section, we conducted a series of experiments on D4RL Gym-MuJoCo benchmark to verify the effectiveness of our proposed method. The experiments aimed to answer the following questions: 1) Can the guidance sampling method effectively guide the learned policy to choose good actions? 2) How does the performance of the proposed RDGP compare with existing state-of-the-art ORL methods on standard ORL benchmarks?

A. Implementation Details

The D4RL benchmark includes three classic Gym-MuJoCo continuous control tasks: Halfcheetah-v2, Hopper-v2, and Walker2d-v2. These tasks simulate scenarios encountered by robots in real industrial systems. As illustrated in Fig. 3, the goal of Halfcheetah-v2 is to apply torque to the joints to make the cheetah robot run forward as fast as possible; the goal of Hopper-v2 is to make hops move forward by applying torques on the three hinges connecting the four body parts; the goal of Walker2d-v2 is to make a robot walk forward by applying torques on the six hinges connecting the seven body parts. For each task, we conducted experiments with four different quality datasets: medium, medium-replay, medium-expert, and expert. Specifically, the medium and expert datasets collected experiences obtained by interacting with the task environment

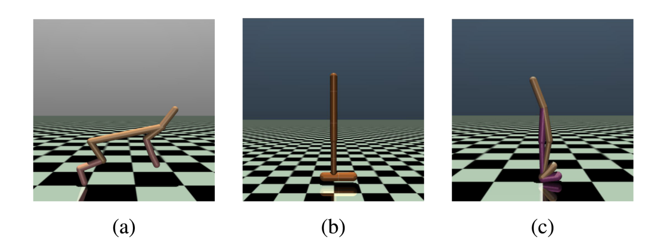


Fig. 3. Illustration of Gym-MuJoCo continuous control tasks. (a) Halfcheetah-v2. (b) Hopper-v2. (c) Walker2d-v2.

TABLE I
Reference Returns of Gym-MuJoCo Tasks

Task	Ref. Expert return	Ref.Random return
HalfCheetah-v2	12135.0	−280.1
Hopper-v2	3234.3	−20.2
Walker2d-v2	4592.3	1.63

for 1 million steps using a medium soft actor-critic (SAC) policy and an expert SAC policy, respectively. The medium-replay dataset collected interaction experiences from an SAC policy trained from the initial stage to medium performance. The medium-expert dataset is a mixture of 1 million samples from the medium dataset and 1 million samples from the expert dataset. For ease of cross-task comparison, we calculate normalized scores (NS) as evaluation metrics, roughly normalizing the score of each environment to a range between 0 and 100. (When the normalized score exceeds 100, it indicates that the learned policy's performance surpasses that of the expert policy.)

$$\text{NS} = 100 \times \frac{\text{Average return} - \text{Ref.random return}}{\text{Ref.expert return} - \text{Ref.random return}} \quad (21)$$

where the Ref.expert return and Ref.random return for each task in the D4RL benchmark are provided in Table I.

We first evaluated the performance of the proposed RBPD and RGDP on all 12 datasets and then compared them with several state-of-the-art ORL methods, including BCQ [10], TD3 with behavior cloning (TD3BC) [19], conservative Q-learning (CQL) [15], and implicit Q-learning (IQL) [17]. BCQ and TD3BC are the most popular policy constraint methods, CQL is the most popular value function penalty method, and IQL is the most popular in-sample method. For a fair comparison, the parameters for all compared methods are kept consistent with the original papers, and the parameters for RBPD and RDGP are listed in Table II. The experimental results are presented in Fig. 4 and Table III. Referring to [34], we set the noise coefficient in the forward diffusion process as $\beta_i = 1 - e^{-\beta_{\min}(\frac{1}{N}) - \frac{2i-1}{2N^2}(\beta_{\max} - \beta_{\min})}$, where $\beta_{\min} = 0.1$ and $\beta_{\max} = 10.0$. It is worth noting that the performance of RDGP is related to the perturbation threshold ϑ. Results for RDGP w/o p (which introduces the RDGP upon RBPD but omits the perturbation component) and RDGP (which use the optimal perturbation threshold) are shown in Fig. 4 and Table III. To show the influence of the perturbation threshold ϑ on the performance of RDGP, we conducted experiments with ϑ values of 0, 0.01, 0.02, 0.1, and 0.2 on four different quality datasets of the

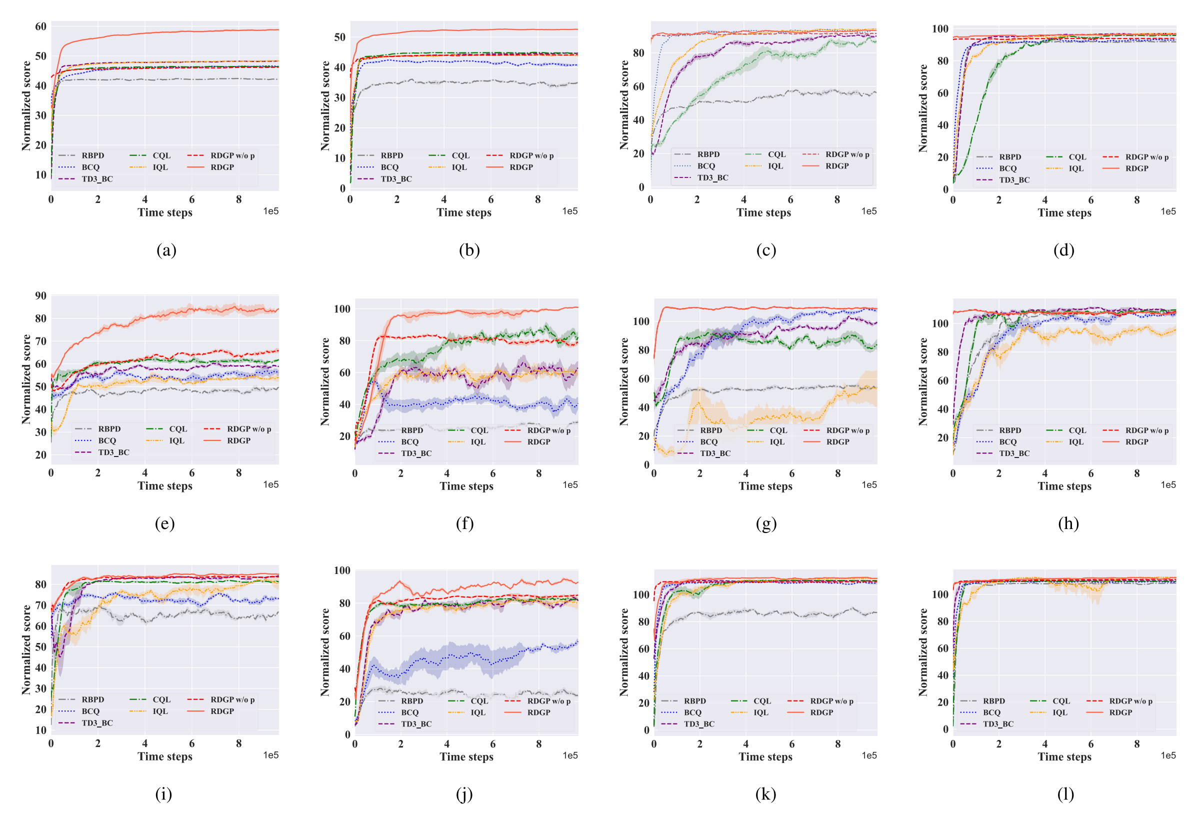


Fig. 4. Normalized score curves of evaluated ORL methods on D4RL Gym-MuJoCo tasks. (a) Halfcheetah-medium-v2. (b) Halfcheetah-medium-replay-v2. (c) Halfcheetah-medium-expert-v2. (d) Halfcheetah-expert-v2. (e) Hopper-medium-v2. (f) Hopper-medium-replay-v2. (g) Hopper-medium-expert-v2. (h) Hopper-expert-v2. (i) Walker2d-medium-v2. (j) Walker2d-medium-replay-v2. (k) Walker2d-medium-expert-v2. (l) Walker2d-expert-v2.

TABLE II
HYPERPARAMETERS OF RBPD AND RGDP

Hyperparameter	Value
Hidden layers of neural networks	2
Hidden dimensions of neural networks	256
Activation function	Mish
Optimizer	Adam
Learning rate	3×10^{-4}
Mini-batch size	256
Discount factor	0.99
Maximum diffusion time step	5
Target network update rate	5×10^{-3}

Halfcheetah-v2 task. The experimental results are presented in the Supplementary Material.

In the experimental process, each method was independently run four times with different random seeds, and each run consisted of training for 1 million steps. Every 5000 steps during the training process, an evaluation was conducted, where the policy interacted with the environment for 10 episodes (with a maximum of 1000 steps per episode). The average return over 10 episodes was used in (21) to calculate the normalized score, which served as the evaluation result. In Fig. 4, the curves and shadows represent the mean and standard deviation of the evaluation results, whereas Table III records the mean (average normalized score) and standard deviation of the last 20 evaluation results as the final performance of the learned policy.

All experiments were conducted on the same device equipped with an NVIDIA GeForce RTX 3090 Ti GPU and an Intel Core i9-12900 K 3.20 GHz CPU.

B. Experimental Results and Analysis

1) Can the Guidance Sampling Method Effectively Guide the Learned Policy to Choose Good Actions?: Comparing the results of RBPD and RDGP w/o p in Fig. 4 and Table III, it can be observed that RDGP w/o p outperforms RBPD across all tasks, particularly showing significant performance improvements in medium-replay and medium-expert datasets. RDGP w/o p solely incorporates the RDGP upon RBPD, omitting the perturbation component. This showcases the efficacy of the RDGP in guiding the learned policy toward optimal action selection. The expert dataset, collected by the expert policy, can achieve a well-performing policy directly by applying RBPD for behavior cloning. RDGP w/o p, built upon this foundation, further improves performance through fine-tuning with the RDGP, resulting in a slight performance enhancement. For the medium-replay and medium-expert datasets collected by mixed behavior policies, RBPD effectively captures the multi-modal distribution of the mixed behavior policy but ultimately only showcases the "average" performance of the mixed policy. In contrast, RDGP w/o p, through guided sampling, directly obtains optimal actions within the support set of the mixed behavior policy, demonstrating the "optimal" performance of the mixed

TABLE III
AVERAGE NORMALIZED SCORE OF EVALUATED ORL METHODS ON D4RL GYM-MUJOCO TASKS

		RBPD	BCQ	TD3BC	CQL	IQL	RDGP w/o p	RDGP
halfcheetah	M	42.1±0.1	46.5±0.3	48.1±0.1	46.1±0.1	48.3±0.1	46.1±0.1	**58.7±0.6**
	MR	34.3±0.8	40.8±1.0	44.6±0.4	44.7±0.2	43.9±0.4	44.1±0.1	**52.5±0.2**
	ME	55.7±0.9	92.7±0.5	90.1±2.3	86.3±3.3	94.0±0.7	91.5±0.3	**95.6±0.5**
	E	91.9±0.4	93.0±1.0	96.9±0.6	95.8±0.3	**96.9±0.3**	93.9±0.2	96.3±0.1
hopper	M	48.9±1.4	56.3±4.6	58.8±1.7	61.0±1.0	53.7±2.0	65.1±0.9	**83.5±3.3**
	MR	25.7±2.1	39.9±9.6	59.6±16.2	81.6±7.6	60.1±7.5	78.7±2.9	**100.3±0.9**
	ME	53.8±2.7	107.7±1.8	99.8±2.1	84.4±9.5	52.7±29.8	109.2±0.8	**109.2±0.7**
	E	106.9±0.9	106.1±2.9	108.8±2.5	**108.8±1.6**	93.4±6.6	107.2±0.4	107.2±0.4
walker2d	M	64.9±2.2	72.7±2.8	83.5±0.9	81.4±0.6	81.6±3.7	83.5±0.4	**85.4±0.4**
	MR	25.2±2.9	53.7±3.9	80.8±2.9	82.5±0.9	80.8±8.2	84.4±2.0	**93.2±1.5**
	ME	86.6±2.5	108.7±1.6	110.2±0.4	110.2±0.3	111.8±0.6	109.5±0.1	**112.4±0.5**
	E	108.2±0.1	109.2±0.2	110.1±0.1	109.6±0.1	**112.6±0.6**	111.1±0.3	111.9±0.7
total		744.2	927.3	991.3	992.4	929.8	1024.3	**1106.2**

Average over last 10% evaluations with four seeds. Simplify Medium=M, Medium-Replay=MR, Medium-Expert=ME, Expert=E.
The values in bold and underlined indicate the optimal and suboptimal indexes obtained by all evaluated methods.

policy. Although the Medium dataset has less data diversity compared with medium-replay and medium-expert, the guided sampling process can still choose better actions, resulting in significant performance improvements.

2) How Does the Performance of the Proposed RDGP Compare With Existing State-of-the-Art ORL Methods on Standard ORL Benchmarks?: As shown in Fig. 4 and Table III, RDGP w/o p achieves the highest average NS on the Hopper-medium-v2, Hopper-medium-replay-v2, Walker2d-medium-v2, and Walker2d-medium-replay-v2 datasets without changing any hyperparameters. On the remaining datasets, RDGP w/o p obtains slightly lower average NS compared with the highest scores obtained by the baselines, but it consistently exhibits lower standard deviations. Thanks to the pretrained diffusion model that represents the behavior policy, the training process of RDGP w/o p is highly stable, and the normalized score curves do not exhibit significant fluctuations. RDGP w/o p achieves the lowest standard deviations in the majority of datasets. The sum of the average NS for RDGP w/o p across all 12 datasets achieves 1024.3, surpassing all baselines. Comparing with BCQ, RDGP w/o p demonstrates a significant performance improvement on the Hopper-medium-replay-v2 and Walker-medium-replay-v2 datasets. BCQ struggles to precisely sample the optimal action from a diverse behavior policy support set, whereas RDGP utilizes Q-values as guidance, consistently sampling actions with maximum Q-values within the support set. TD3BC employs distance metric regularization during the policy improvement process to achieve policy constraints, forcing the density distribution of learned policy to be close to that of the behavior policy. However, this approach results in suboptimal performance on the medium and medium-replay datasets. CQL and IQL achieve relatively high average NS, but their training processes are unstable, as evidenced by significant fluctuations in the normalized score curves on the Hopper-medium-replay-v2 and Hopper-medium-expert-v2 datasets.

Adding proper perturbation to RDGP achieved the highest average NS on most test datasets, especially outperforming existing state-of-the-art methods on the medium and medium-replay datasets. For the medium and medium-replay datasets, where the optimal actions are likely outside the support set of behavior policy, RDGP with perturbation effectively generalizes to OOD regions, allowing the learned policy to surpass the performance limitations of the behavior policy. Although the average NS of RDGP on the Halfcheetah-expert-v2 and Hopper-expert-v2 datasets are not optimal, it still achieves competitive performance with a lower standard deviation. The sum of the average NS across all 12 datasets for RDGP achieves 1106.2, significantly higher than the existing state-of-the-art methods compared, demonstrating the superiority of the proposed method.

V. CONCLUSION

To alleviate the problem of distribution shift in ORL, this article proposes a novel ORL method based on RDGP. RDGP utilizes the reverse diffusion process in diffusion model to represent the behavior policy, and then trains the RDGP to generate actions from the reverse diffusion process, choosing actions that maximize the Q-value within the support set of behavior policy. RDGP avoids the need for multiple sampling processes found in previous methods, allowing a flexible selection of actions from the support set of behavior policy without being influenced by the density distribution of behavior policy. When facing low-quality datasets, RDGP further introduces trainable perturbations to the generated actions, enabling the learned policy to escape the performance limitations of behavior policy. Comparative experiments on the D4RL benchmark with various state-of-the-art ORL methods demonstrate that RDGP achieves competitive performance across all tasks. However, since each sampling action of RDGP requires a reverse diffusion process, its decision-making speed is slower compared with baselines. This characteristic makes RDGP more suitable for control tasks requiring high precision but without fast response. In future work, we will consider incorporating accelerated sampling methods such as denoising diffusion implicit model (DDIM) [35] and diffusion probabilistic model (DPM)-solver [36] in diffusion models to improve the decision-making speed of RDGP.

REFERENCES

[1] L. Chen et al., “Transformer-based imitative reinforcement learning for multirobot path planning,” *IEEE Trans. Ind. Inform.*, vol. 19, no. 10, pp. 10 233–10 243, Oct. 2023.

[2] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[3] S. Fujimoto, E. Conti, M. Ghavamzadeh, and J. Pineau, "Benchmarking batch deep reinforcement learning algorithms," 2019, *arXiv:1910.01708*.
[4] H. Park, D. Min, J. h. Ryu, and D. G. Choi, "DIP-QL: A novel reinforcement learning method for constrained industrial systems," *IEEE Trans. Ind. Inform.*, vol. 18, no. 11, pp. 7494–7503, Nov. 2022.
[5] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
[6] L. Zhang, M. Zhou, and Z. Li, "An intelligent train operation method based on event-driven deep reinforcement learning," *IEEE Trans. Ind. Inform.*, vol. 18, no. 10, pp. 6973–6980, Oct. 2022.
[7] X. Wang, J. Zhang, D. Hou, and Y. Cheng, "Autonomous driving based on approximate safe action," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 12, pp. 14 320–14 328, Dec. 2023.
[8] Z. Yu and X. Zhang, "Actor-critic alignment for offline-to-online reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 40 452–40 474.
[9] B. Hu, Y. Xiao, S. Zhang, and B. Liu, "A data-driven solution for energy management strategy of hybrid electric vehicles based on uncertainty-aware model-based offline reinforcement learning," *IEEE Trans. Ind. Inform.*, vol. 19, no. 6, pp. 7709–7719, Jun. 2023.
[10] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2052–2062.
[11] Y. Cheng, L. Huang, C. L. P. Chen, and X. Wang, "Robust actor-critic with relative entropy regulating actor," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 11, pp. 9054–9063, Nov. 2023.
[12] Y. Wu, G. Tucker, and O. Nachum, "Behavior regularized offline reinforcement learning," 2019, *arXiv:1911.11361*.
[13] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, "Stabilizing off-policy Q-learning via bootstrapping error reduction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 11 784–11 794.
[14] Y. Ran, Y. C. Li, F. Zhang, Z. Zhang, and Y. Yu, "Policy regularization with dataset constraint for offline reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 28 701–28 717.
[15] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-learning for offline reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1179–1191.
[16] I. Kostrikov, R. Fergus, J. Tompson, and O. Nachum, "Offline reinforcement learning with fisher divergence critic regularization," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 5774–5783.
[17] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit Q-learning," 2021, *arXiv:2110.06169*.
[18] Z. Wang et al., "Critic regularized regression," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 7768–7778.
[19] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 20 132–20 145.
[20] Y. Mao, H. Zhang, C. Chen, Y. Xu, and X. Ji, "Supported trust region optimization for offline reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 23 829–23 851.
[21] S. K. S. Ghasemipour, D. Schuurmans, and S. S. Gu, "EMaQ: Expected-max Q-learning operator for simple yet effective offline and online RL," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 3682–3691.
[22] J. Li, E. Zhang, M. Yin, Q. Bai, Y. X. Wang, and W. Y. Wang, "Offline reinforcement learning with closed-form policy improvement operators," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 20 485–20 528.
[23] Q. Vuong, A. Kumar, S. Levine, and Y. Chebotar, "Dual generator offline reinforcement learning," 2022, *arXiv:2211.01471*.
[24] H. Chen, C. Lu, C. Ying, H. Su, and J. Zhu, "Offline reinforcement learning via high-fidelity generative behavior modeling," 2022, *arXiv:2209.14548*.
[25] P. Dhariwal and A. Nichol, "Diffusion models beat GANs on image synthesis," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 8780–8794.
[26] Y. Song, J. S.-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," 2020, *arXiv:2011.13456*.
[27] Z. Wang, J. J. Hunt, and M. Zhou, "Diffusion policies as an expressive policy class for offline reinforcement learning," 2022, *arXiv:2208.06193*.
[28] B. Kang, X. Ma, C. Du, T. Pang, and S. Yan, "Efficient diffusion policies for offline reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 67195–67212.
[29] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 6840–6851.
[30] F. Croitoru, V. Hondru, R. T. Ionescu, and M. Shah, "Diffusion models in vision: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 9, pp. 10 850–10 869, Sep. 2023.
[31] C. Luo, "Understanding diffusion models: A unified perspective," 2022, *arXiv:2208.11970*.
[32] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
[33] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.
[34] Z. Xiao, K. Kreis, and A. Vahdat, "Tackling the generative learning trilemma with denoising diffusion GANs," 2021, *arXiv:2112.07804*.
[35] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," 2020, *arXiv:2010.02502*.
[36] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu, "DPM-solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 5775–5787.



Jiazhi Zhang (Graduate Student Member, IEEE) received the M.S. degree in control science and engineering from China University of Mining and Technology, Xuzhou, China, in 2023. He is currently working toward the Ph.D. degree in control science and engineering with the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou.

His main research interests include offline reinforcement learning and safe reinforcement learning.



Yuhu Cheng (Member, IEEE) received the Ph.D. degree in control science and technology from Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2005.

He is currently a Professor with the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, China. His main research interests include machine learning and intelligent system.



Shuo Cao (Graduate Student Member, IEEE) received the M.S. degree in control science and engineering from Jiangsu University, Zhenjiang, China, in 2021. He is currently working toward the Ph.D. degree in control science and engineering with the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, China.

His main research interests include offline reinforcement learning and optimal control.



Xuesong Wang (Member, IEEE) received the Ph.D. degree in control science and technology from China University of Mining and Technology, Xuzhou, China, in 2002.

She is currently the Dean with the Engineering Research Center of Intelligent Control for Underground Space, Ministry of Education, Xuzhou, and a Professor with the School of Information and Control Engineering, China University of Mining and Technology Xuzhou. Her main research interests include machine learning and artificial intelligence.

Dr. Wang is currently an Associate Editor of IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, IEEE SIGNAL PROCESSING LETTERS, *International Journal of Machine Learning and Cybernetics*, *Acta Automatica Sinica*, and *Acta Electronica Sinica*.